

Platform Support for Pedagogical Scenarios

Yvan Peter, Thomas Vantroys

▶ **To cite this version:**

Yvan Peter, Thomas Vantroys. Platform Support for Pedagogical Scenarios. Journal of Educational Technology and Society, 2005, 8(3), pp.122-137. hal-00190116

HAL Id: hal-00190116
https://telearn.hal.science/hal-00190116

Submitted on 23 Nov 2007

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Platform Support for Pedagogical Scenarios

Yvan Peter and Thomas Vantroys
Laboratoire TRIGONE, Université de Lille 1
Bât B6, Cité Scientifique
59655 Villeneuve d'Ascq cedex - France
Yvan.Peter@univ-lille1.fr
Thomas.Vantroys@univ-lille1.fr

ABSTRACT

This article deals with providing support for the execution of pedagogical scenarios in Learning Management Systems. It takes an engineering point of view to identifies actors, design and use processes. Next it defines the necessary capabilities of a platform so that actors can manage or use pedagogical scenarios. The second part of the article is dedicated to the presentation of the design and properties of a component we have developed for the execution of scenarios: a flexible Workflow engine and a user interface providing indicators about the student progress within a unit of study.

Keywords

Pedagogical scenario, Unit of study engineering, Scenario execution, Workflow engine

Introduction

E-Learning has been spotted as the solution to the growing needs of education. Many institutions and enterprises have set up Learning Management Systems (LMS) and organized their work around these new technologies. However, when the number of learners grow, it becomes difficult for tutors to support them correctly while still being aware of individuals that need some special care because they experience difficulties. Thus, LMS should provide some level of automation for the management of the activities within units of study while keeping tutors aware of students having difficulties and permitting an adaptation of the activities to these special cases. One way to define the activities that will take place within a unit of study is to describe them in a pedagogical scenario. Such a scenario defines the activities which must be done by the learners and the tutors, the sequencing of these activities as well as the learning objects and tools that should be provided to the different actors. For instance, the emerging standard IMS-LD (IMS, 2003) uses a theatrical metaphor where the activities take place in different acts that define the sequencing. The activities are associated to roles corresponding to users and to an environment composed of learning objects and tools. The benefit of the use of pedagogical scenarios, as stated in Hummel *et al.* (2004), is that the focus is put on the learning activities that should be done to achieve a learning objective rather than the learning objects. In this approach, the activities provide a context for the use of the learning objects rather than having a "passive" consumption of them. This article presents our work towards providing support for pedagogical scenarios at both organizational and technical level. Indeed, the production and use of pedagogical scenarios should be backed by a correct organization as it is the case for producing and using learning objects. For this reason, we have defined the life cycle of a pedagogical scenario and identified the different actors involved in it. Considering the execution, we have taken care of flexibility issues to increase adaptability to the learners and reusability, i.e., with have the possibility to modify the scenario at run time. This is useful because it is sometimes difficult to know beforehand the activities that will take place (e.g., in life long learning where the learning process lasts a long time) or because an adaptation of the course is needed (e.g., to suit the learners' level).

The first part of the article presents the design and use of pedagogical scenarios from a platform engineering point of view. In the second part, we review the requirements towards platforms support for scenario execution. The third part presents the basis we have used to build our solution. The fourth part is dedicated to the *Cooperative Open Workflow*, the solution we have developed based on Workflow technology and shows how we meet the defined requirements with an illustration on a sample scenario. The solution is compared to existing alternatives. Finally, we present a user interface taking benefit of the Workflow engine to give learners indicators about their progress within a unit of study to enable better planning of their work. The conclusion summarizes the work done and presents ongoing work.

Engineering of pedagogical scenarios

The architecture and use of Learning Management Systems is getting more and more complex. The design and delivery of a new course implies many tasks which rely on highly specialized actors during the whole life cycle.

ISSN 1436-4522 (online) and 1176-3647 (print). © International Forum of Educational Technology & Society (IFETS). The authors and the forum jointly retain the copyright of the articles. Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear the full citation on the first page. Copyrights for components of this work owned by others than IFETS must be honoured. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from the editors at kinshuk@ieee.org.

In the next sections, we will present the design process, the use process as well as the actors involved in the definition and operation of a course in the form of a pedagogical scenario.

Design process

The design process of a course shown in the figure 1 is composed of 6 main phases. This is inspired from the Rational Unified Process (Jacobson *et al.*, 2000). Each phase enables to go to the next or to go back to the previous one if problems related to it appear.

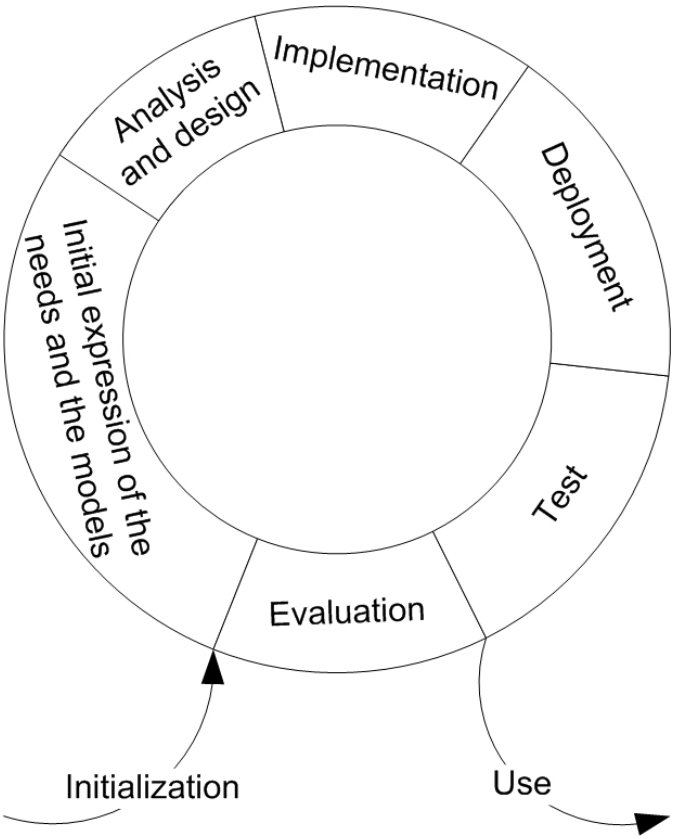


Figure 1. Design process of a unit of study

- **Initial expression of needs and models**. This phase corresponds to the initial definition of a course. A *teacher* will describe in an informal manner the activities of the course and will define the objectives, the prerequisites ... He will also define the type of learning objects and tools which will be necessary. RUP relies on UML use cases for this but there is not any equivalent formalism in the field of Technology Enhanced Learning.
- **Analysis and design.** Following the informal description, a *pedagogical engineer*, which is aware of the technical characteristics of the target platform, will collaborate with the teacher to translate the scenario into a more formal one where the different phases are scheduled. Some of the activities may have to be adapted or replaced depending on the capabilities of the platform. This design phase may require interactions with *pedagogical resource providers* and *component developers*. The former will define the learning objects which are available or have to be created. The latter is more concerned with the development of new software components on the platform to provide the needed tools.
- **Implementation.** This phase corresponds to the development of the software components and pedagogical resources needed for the course. It is mainly done by the component developer in collaboration with the pedagogical engineer. It may also involve a *component assembler* if a functionality can be provided by an existing set of components.
- **Deployment.** All elements developed or assembled during the previous phase are installed on the target platform. This phase concerns the component assembler and the *platform administrator*. At the end of this phase the new course can be referenced.

- **Test.** This phase corresponds to the control of the behaviour of the platform and resources and the validation of the coherence of the model. The tests are done by the pedagogical engineer who is qualified for the control at both the pedagogical and technical level. The component developer and assembler are responsible for the control of the respect of the specifications at the technical level.
- **Evaluation.** During the use of the course model, it is important to monitor the behaviour of the *learners* and *tutors* to see if the initial model is correctly executed. If not, this is the opportunity to reengineer the model to better suit the real use.

At a higher level, there is also a phase for building a learning path composed of many courses which is at the limit between the design and use processes. In this phase, the pedagogical engineer in collaboration with teachers responsible for the units and a *learner orientation adviser* will create a complete learning path.

Use process

The use process describes the different phases corresponding to the life cycle of a course model. These phases are described hereafter:

- **Instantiation.** This phase determines which people will actually take part in the course according to the model. This corresponds to the following roles: *learner*, *tutor* and *referent*. The learner orientation adviser may be involved in the definition of the groups of learners while the tutor is responsible for checking that resources (i.e., learning objects and tools) are available and that the course can begin.
- **Execution.** This phase corresponds to the most visible part where users are involved in a learning process through the platform. The learners will process through the different activities of the course while the tutor checks the correct execution of the model. The referent is responsible for a group and will serve as central point to solve learners' problems be it technical, pedagogical or administrative.

Roles involved in distance learning platforms

So as to define the different tasks implied in operating a course on a Learning Management System, we have defined roles which are related to these tasks. This is done in the same spirit as in the J2EE roles (component developer, assembler ...) (Matena *et al.*, 2001), in design processes (Jacobson *et al.*, 2000) and in application development (Schmidt-Wesche, 2003). The role list may not be exhaustive but rises from our experience and research in the field of distance learning. The different roles are shown figure 2 with their type of responsibility: administration, modelling, technical, use. It must be noted that multiple roles may be taken by the same person even though it is likely that many people will be required due to the increasing complexity of e-learning platforms. We will give a brief description of each role and their relations hereafter.

Teacher

The teacher is the starting point in the creation of a course. He defines, in terms of competencies, the prerequisites and objectives of the pedagogical activities. He may associate specific services (e.g. asynchronous communication), tools (e.g., simulator) or learning objects. The teacher is an expert in a specific domain. He collaborates with the pedagogical engineer to adapt his pedagogical scenario to the capabilities of the platform. He may assist the learner orientation adviser in assessing the competencies of a learner for a particular course.

Pedagogical engineer

The pedagogical engineer is concerned with the whole design process since he has both pedagogical competencies and knowledge of the used platform. He can manage the link between the design of courses, associated developments and use. He helps the teacher formalize his pedagogical scenario according to the capabilities of the platform and can interact with the component developer to introduce new tools or services into the platform. He can guide the learner orientation adviser in designing a new learning path. He can interact with the referent and tutor to handle modifications of a course model.

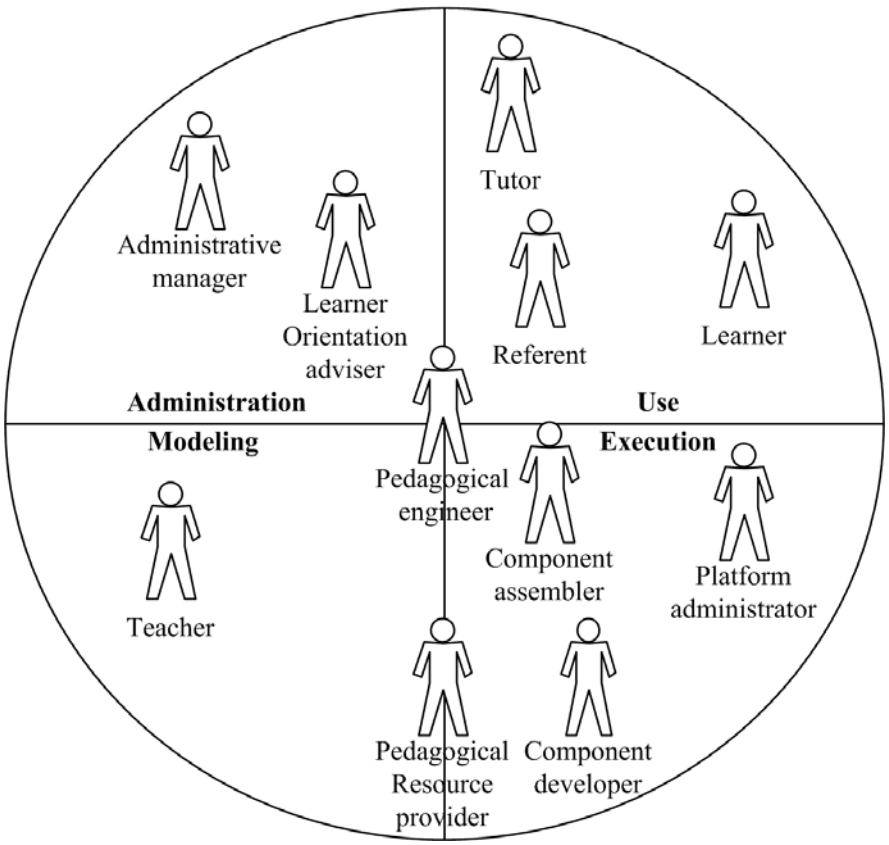


Figure 2. Roles and types of responsibility

Pedagogical resource provider

He designs and develops learning objects for the teacher and pedagogical engineer.

Learner

The learner uses the platform to get new competences. He may work alone or in a group. The learner uses the platform tools and accesses the learning objects in the scope of the assigned activities to learn. He can interact with the tutor to get help on specific topics of the course and with the referent for more general problems. He can build his learning path with the help of the learner orientation adviser.

Tutor

The tutor takes care of the good unfolding of the course and handles the animation of the groups and activities. He follows and assists the learners so as to meet the learning objectives. He informs the referent of any particular difficulties. The tutor can make local modifications to the course model in collaboration with the pedagogical engineer and the referent.

Referent

The referent manages one or more groups. He serves as a unique access point during the whole learning path. He can act as a mediator between the tutors or tutors and learners. He can ask for modifications of the model to the tutors.

Administrative manager

This role is more concerned with administrative management of learning path: enrolment, control of the presence ... He is responsible for the respect of organizational rules and laws.

Learner orientation adviser

The learner orientation adviser welcomes learners and guides them towards the best courses and learning paths according to their competencies and wishes.

Component assembler

The component assembler gathers existing components to create new services or tools for the platform.

Component developer

The component developer is an expert in the development for the target platform and provides basic elements for the component assembler and pedagogical resource provider.

Platform administrator

The platform administrator is responsible for the correct operation of the platform. He manages user accounts in collaboration with the administrative manager, the pedagogical engineer and the referent.

Platform support for pedagogical scenarios

When it comes to the point of deploying a pedagogical scenario on a specific platform, most of the time it comes down to organizing the access to the pedagogical resources and most of the scenario is lost. For instance, if an order has been defined for accessing the resources, this may not be enforced by the platform. This is the case for example with the *OpenUSS* platform (Grob *et al.*, 2004) where one can organize access to learning materials into semesters, and courses but would have to manually deliver materials if he wishes a better control on the order of access. This is mainly due to the fact that the focus has been directed to accessing the resources rather that scripting of the access. In this part, we will describe the properties needed to fully support pedagogical scenarios and how they relate to the roles involved in the operation of a distance learning platform.

Formal model for the pedagogical scenarios

To take advantage from a platform support for pedagogical scenarios, there must be a formal description that can be handled by the platform. Educational Modelling Languages such as IMS-LD (IMS, 2003) can be a good candidate for this. This formal language will be used by the pedagogical engineer to translate the ideas of the teachers and learner orientation advisers while taking into account the concerns of the administrative manager. To this end, a general modelling language should support:

- **Individual and group scheduling of activities.** People can engage in a course or a learning path either individually or within a group. In the latter case, it is reasonable to allow for individual progression between the activities so that each one can progress at his own rhythm. Group synchronization will be enforced only for collaborative activities or to respect time constraints. The definition of the individual parts is done by the teacher and pedagogical engineer.
- **Time management.** It is important to be able to enforce time constraints on a unit of study or learning path. Firstly to support the directives from the administrative manager (e.g., to respect academic years) and secondly for pedagogical reasons (e.g., to limit the duration of an online test).
- **Collaboration support.** Collaborative learning is a means to enhance the learning experience and a way to sustain the learners' motivation (Eales *et al.*, 2002). As such it should be supported.

Platform support for model execution

Platform support for model execution is tightly related to what can be expressed in the model. In a general manner, the platform is responsible for automating the progress of the learners from activity to activity within a course while providing the right tools and learning objects. This capability is of paramount importance to relieve the tutors from a daunting task in case of large groups of students.

Monitoring of the model execution

Automation of the learning process is a great help in the operation of distance learning. However, it should be accompanied with monitoring facilities to assist the tutors and referents in the follow-up of the learners. The platform should provide the means to raise awareness of the potential problems that learners may have (e.g., slow progression within a group, violation of a time constraint ...).

Enhanced management of the models

Having identified a learner or group problem thanks to the monitoring facilities, the tutor and pedagogical engineer should be able to modify the model at run time to achieve the pedagogical objectives. Moreover, if the reengineered model is better, the modified model should be available for the next execution. This ensures that the pedagogical scenario takes benefit from the previous experiences. One can also find easier to build courses from existing activities or scenarios.

Planning support

From the learner point of view, the engine should provide means to assess his progression within a course and relative to the group. This is another kind of monitoring facility. It should also assist the learner in planning his activities by showing the activities to perform to complete the course.

Building flexible support for pedagogical scenario

In our research work on supporting the execution of pedagogical scenarios, we have decided not to focus on a specific platform but rather to develop a dedicated component to be embedded in existing Learning Management Systems. This has raised some technical issues to bear in mind during the design of this component:

- **Standard based**: Since we are designing a technical component to be embedded in different platforms, the first requirement is that we have is to keep up with existing standards.
- **Integration support**: Integration within a platform should be easy, for this Web Services standards and event based communication are good solutions.
- **Persistence**: Learning scenarios may correspond to long running processes. For this reason, we have to take care of persistence issues.

To tackle the last two issues, we have chosen to rely on the J2EE standards. The component has been developed using Enterprise JavaBeans components (Eberhart & Fischer, 2002) that deal with the persistence issues. These components can be accessed through RMI/IIOP or as Web Services so we are able to support integration with RMI, CORBA and SOAP protocols.

Considering the first issue, standards, we have headed towards a general solution rather than using a specific pedagogical language. Executing a pedagogical scenario can be compared to the execution of a traditional process (e.g., in business, administration ...). The natural solution for process execution is a Workflow engine. However, traditional Workflow engines are too strict to support our requirement for model modification at run time. In general we have not found any that can handle all the requirements we have identified in the previous section. For this reason we have started to build up our own solution starting from existing Workflow standards: Workflow Management Coalition standards and the Object Management Group's Workflow Management Facility. An additional benefit of this approach is that we are not restricted to the description of pedagogical scenarios but we can handle more administrative parts of the platforms such as registration, group creation... In the next section we will describe the standards we have used before presenting how we have fulfilled our requirements.

Workflow standards

The Workflow Reference Model and XML Process Definition Language

The *Workflow Reference Model* (Hollingsworth, 1995) is a standard from the Workflow Management Coalition (WfMC) whose aim is to promote the use of Workflow through standardization and interoperability. This standard does not define the Workflow engine itself but rather its interfaces. In this article, we are more concerned with interface 1 which defines an XML language called XPDL (*XML Process Definition Language*) (WFMC, 2002) to define process models regardless of the enacting platform. Interface 5 is also of interest since it defines how administration and monitoring tools can interact with the Workflow engine.

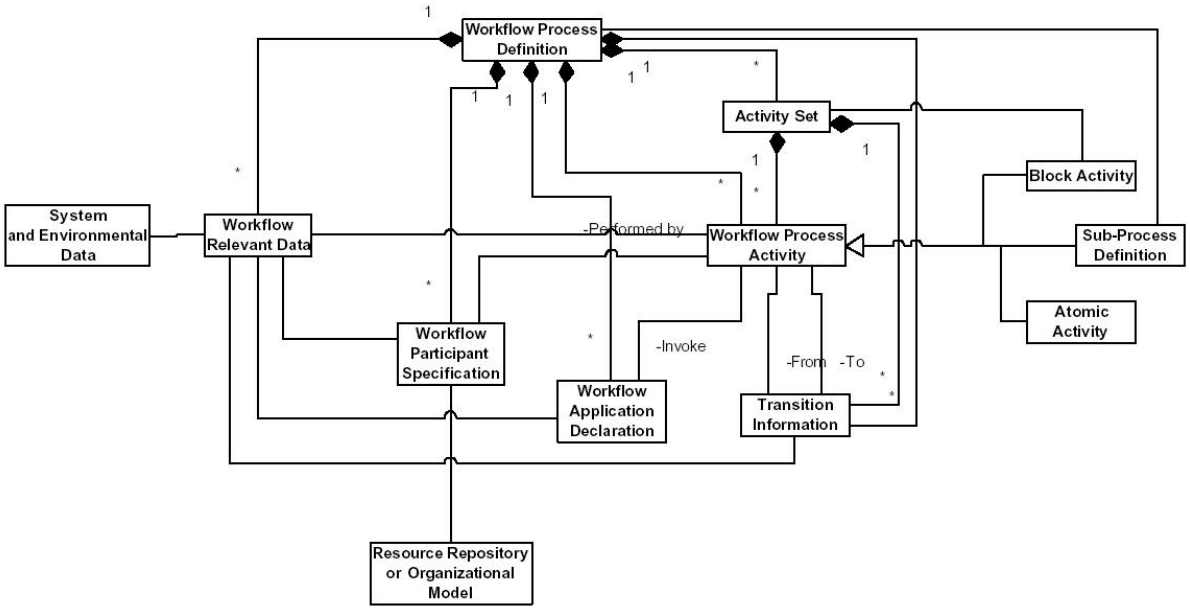


Figure 3. XPDL Meta-model from (WFMC, 2002)

Figure 3 shows the meta-model of the XPDL language with which we can define a pedagogical scenario for execution on a Workflow engine. The 6 top-level entities are *Process*, *Activity*, *Transition*, *Application*, *Relevant Data* and *Participant*.

- **Process** defines the way to achieve a common goal, i.e., the path between the different activities. It determines the execution context (overall description, input values …). This element is the container of all the other entities of the metamodel. This corresponds to the unit of study level;
- **Activity** defines the work to realize. There are three types of activities. Sub-flow activity allows executing another sub-process, block activity consists of an activity set which is an aggregation of activities and transitions and atomic activity is the real work to do. At the execution time, this work is transformed into work items which are executed by participants and/or applications. This defines the tasks which have to be done by the different roles: learner, tutor...
- **Transitions** are the links between different activities. They define the control flow inside the process. This will define how activities are scheduled within the pedagogical scenario;
- **Application** corresponds to the applications which can be used to perform the activities. There, we can define the tools used by the learners and support staff (e.g., mail, forum, text editor ...);
- **Relevant Data** are the data used and produced by the process and the activities. This data can be linked with the data managed by the learning management system;
- **Participant** represents a human, role or group to who work items are assigned. Participant can be linked with the user accounts and groups as defined on the LMS.

We have used the XPDL language with extensions to support the definition of group activities and time constraints. We have also separated process and activity models into two parts so as to be able to build a scenario from existing activities.

Workflow Management Facility

The *Workflow Management Facility* (WMF) (OMG, 2000) standardizes the architecture of the Workflow engine. It has been defined by the Object Management Group (OMG) in accordance with the reference model of the WfMC. This standard defines the interfaces of objects such as *WfProcess* and *WfActivity* or *WfResource* which correspond to the elements described in a XPDL process model. There is also a hierarchy of **EventAudit* interfaces which corresponds to the events tracked by the Workflow engine. This is useful to build a monitoring facility. We have made an implementation of the interfaces proposed in the WMF with extensions to support collaborative activities, group management and to enhance the reuse of models both at the process and activity levels.

Cooperative Open Workflow

In this section we will describe the Workflow engine we have developed based on these standards which is called *Cooperative Open Workflow* (COW), (Vantroys & Peter, 2003). We will first illustrate how we can model a sample pedagogical scenario. We will next focus on collaborative activities and time constraints. Then we will show how the model is handled by the Workflow engine at run time. Finally, we give some details about the implementation before presenting related work.

Unit of study modelling

The main function of the Workflow system is to schedule the activities of a course. Such a course is attended by a group of learners (individual learning being a special case since the "group" is reduced to one learner). In the sequel, we will take a course in physics as an example of the modelling of a course to illustrate the management of models and instance inside the Workflow and the management of both individual and group activities. The scenario corresponding to the course is composed of four activities described hereafter:

- *Course learning* activity associated to the role *learner* which will be given access to a set of learning objects related to the subject. We can set a maximum duration to this activity so that learners do not take too long on this activity;
- *Self test* activity associated to the role *learner* which will have to pass a multiple-choice questionnaire. Since this is a test, we would like to state that it has to be performed in a limited time;
- *Test correction* activity associated to the role *tutor* which will review the results of the tests and look at the errors made by the students;
- *Discussion about the unit of study* activity associated to the role *learner* and *tutor*. Here the tutor can initiate a discussion about the concepts studied based on the tests results. We can set a deadline for starting this activity to help synchronize everyone.

Since some parts of the course can be realized at his own rhythm by each learner, one has to take it into account so as to enable flexibility in the schedule of the activities. There are two ways to manage the schedule of the activities for a group of learners:

- In the first mode, an activity is terminated only when all the learners have terminated it. In such a way, the activities of a whole group are synchronized. Even though it is very close to traditional face to face learning, it does not take benefit of distance learning mode;
- The second mode identifies the parts of a course that can be realized autonomously. This way each learner can progress at his own rhythm inside a group with some activities giving a synchronization point to the group.

These two modes are supported in COW by means of sub-processes. In our scenario, the teacher decides that the three first activities can be realized individually by each learner. These activities are then modelled into a single process. The process corresponding to the whole course is then composed of two sequential activities (see figure 4), the first one being in fact a reference to the individual work sub-process and the second one corresponding to a synchronous discussion between the members of the group.

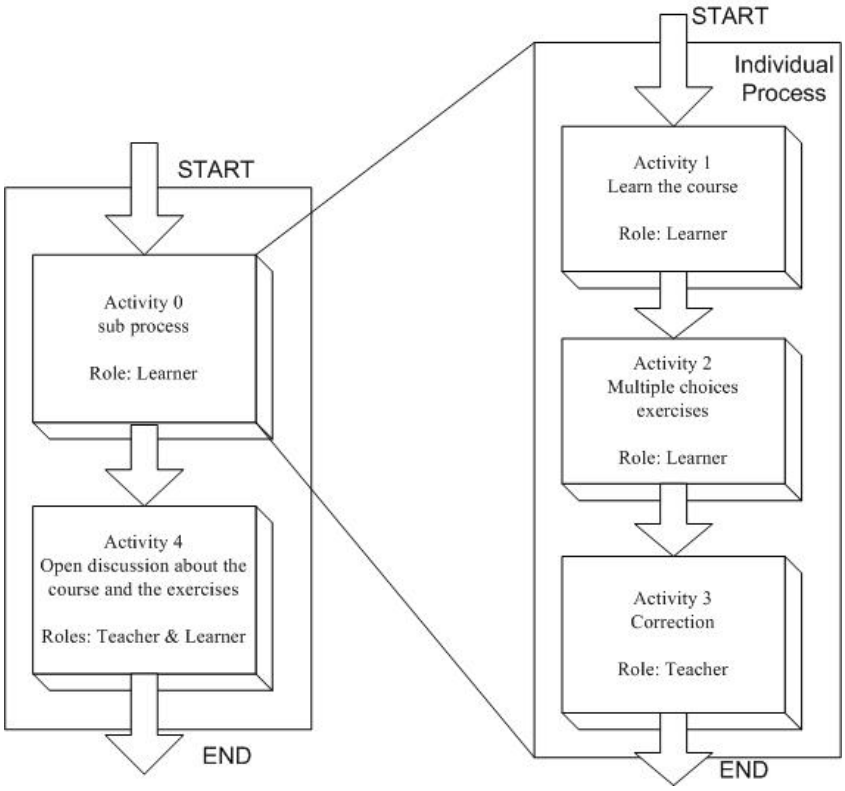


Figure 4. Pedagogical scenario model

Figure 5 shows the model of the global course model in the XML format (derived from XPDL). We have separated the process and activity models. This way, we can easily change and replace an activity by another or modify an activity within a process without changing the whole schedule. We can also build a new process starting from individual existing activities. The model presented defines the identifiers of the activity models that compose the process and the transitions that order these activities. Figure 6 shows the activity model of the discussion referenced in the global process.

```
<WorkflowProcess Id="physics"
                 name="physics">
  ...
  <Activities>
    <ActivityLink Id="START"
                  URL="start" />
    <ActivityLink Id="A1"
            URL="read_and_self_test"/>
    <ActivityLink Id="A2" URL="chat"/>
    <ActivityLink Id="END" URL="end"/>
  </Activities>
  <Transitions>
    <Transition Id="T0"
                From="START" To="A1" />
    <Transition Id="T1"
                From="A1" To="A2" />
    <Transition Id="T2"
                From="A2" To="END" />
</WorkflowProcess>
```

Figure 5. XML model for the course

Collaborative activities

To handle collaborative activities, we have made some modifications to XPDL and the WMF to introduce the notion of *workitem*. A workitem is an atomic piece of work and an activity is composed of workitems and defines the execution context for the inner workitems. In the simplest case, there is only one workitem in an activity. However, within a collaborative activity, there can be more workitems. A workitem is attributed to a role. So if multiple actors have the same role, there will be an instance of the workitem for each of them in the activity. Resources are allocated to the workitems rather than the activity.

```
<Activity Id="chat" Name="chat" />
...
  <Implementation>
    <Tool Id="chat_tool" />
  </Implementation>
  <Group><YES/></Group>
  <Performer>
    <Participant Id="R1"
                 Name="Learner"
                 ParticipantType="ROLE" />
    <Participant Id="R2
                 Name="Tutor"
                 ParticipantType="ROLE" />
  </Performer>
  ...
</Activity>
```

Figure 6. XML model of a collaborative activity

In our example the discussion activity (figure 6) could be done with a chat tool. During this activity, each people having the role learner or tutor will have a workitem corresponding to the work to do. Here, everyone has the same task since no specific workitems have been defined. However, the learners and the tutor may not have the same rights on the tool since they do not have the same roles. There is only one tool defined for this activity which is not actually identified in the model to enable late binding depending on the platform and available tools.

Time constraints

Management of time constraints is an important aspect of learning activities. This is particularly true for group based learning where there must not be too much lag between the learners. COW supports the notion of *deadlines* which correspond to the time at which an activity must be started or completed. It also supports the notion of *limit* which defines the minimal or maximal duration of an activity. When a stop deadline or a maximum duration limit is reached, the Workflow engine suspends the activity. It can then use different policies to handle this case. For example, the system can terminate the activity authoritatively or notify the tutor who will take a decision about it. These behaviours can be dynamically changed at run-time by the tutor to be adapted to a specific context.

Our scenario defines two types of time constraints. The first one is an end deadline for accessing learning objects which is illustrated figure 7. Here the end deadline is set to four days (not including week ends) and the strategy is to send a mail to the student to warn him that he should proceed to the next activity.

```
<Deadline>
  <EndDeadLine>
    <AdjustedTime>
      <Date day="4"/>
      <strategy id=WithoutWeekEnd/>
    </AdjustedTime>
    <Strategy id="sendMail">
  </EndDeadLine>
</Deadline>
```

Figure 7. XML model of deadline constraint

The second time constraint is defined in the self-test activity, the maximum duration will be 3 hours as illustrated in figure 8 and when this duration is reached, the strategy will be to automatically complete the activity.

```
<Limit>
  <MaxTime>
    <AdjustedTime>
      <Date hour="3"/>
    </AdjustedTime>
    <Strategy id=automaticCompletion/>
  </MaxTime>
</Limit>
```

Figure 8. XML model of duration constraint

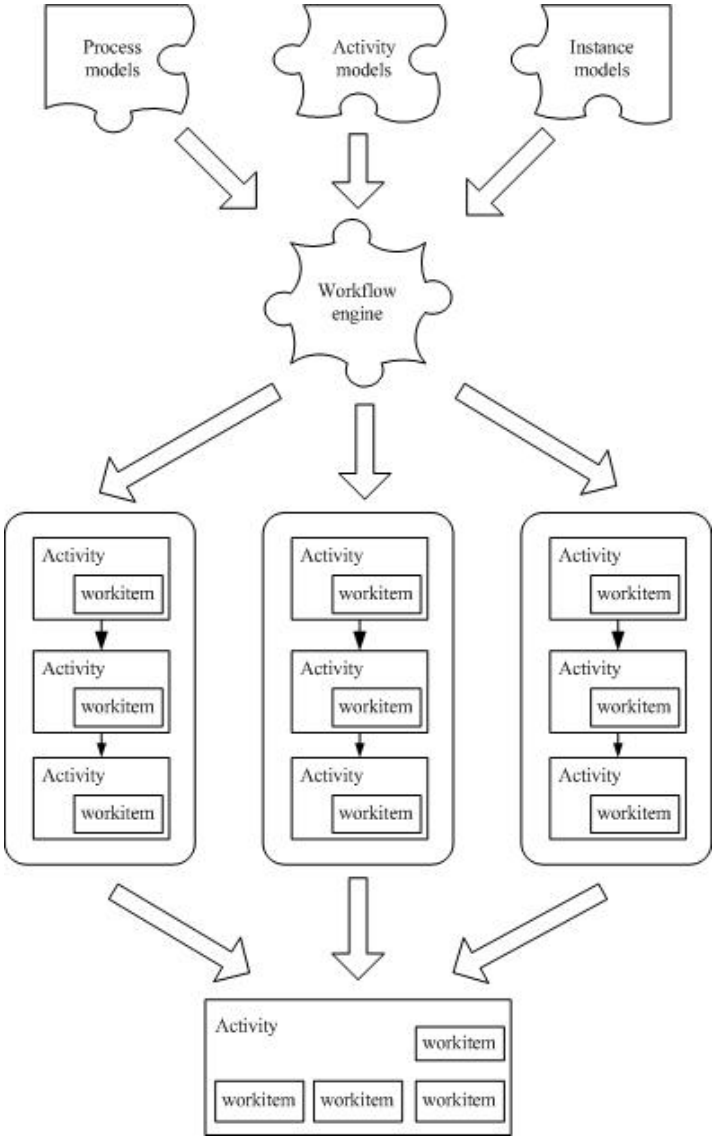


Figure 9. Pedagogical scenario model instantiation

Course instance

The creation of a process instance requires an instance model which describes the mapping of roles to actual users and of resource identifiers to actual learning objects and tools. The mappings can be global to the model or defined on an activity basis. This separation between process model and instance data allows a better reuse of models. Figure 9 illustrates the Workflow operation for a group of three learners. Taking the process and activities models and instance data, the Workflow engine will create a sub process for each learner. These sub

processes contain the three activities that can be performed individually by each learner and each activity contains only one workitem. The third activity performed by the tutor role will have three workitems assigned from different processes. When all sub processes are terminated, the engine will create a collaborative activity with one workitem for each learner and one for the tutor.

Implementation of the Workflow engine

The implementation of the Workflow has been done following a micro-kernel architecture. The basic functions related to the scheduling of the activities constitute the kernel and corresponds to the implementation of the WMF. We can then add more elaborated services on top of the kernel following a facade pattern (Gamma et *al.*, 1995). For instance one facade can build the workitem list of a user. This type of facade is dedicated to the "normal" use of the engine.

Flexibility is provided following the Meta-Object Protocol (Kiczales *et al.*, 1991). This means that some facades are dedicated to the consultation and modification of the process model and engine behaviour:

- Process modification consists in adding/deleting activities and transitions or modifying the activity model (e.g., changing tools or learning objects, modifying workitem assignment). These changes can be realized for one learner or for a group of learners.
- Behaviour modification changes the way the model is interpreted. It has been particularly used for time constraints and exception handling. For instance duration can be interpreted in absolute time or taking work days into account. One can define different strategies and use them following the strategy pattern (Gamma *et al.*, 1995).

The integration within Learning Management Systems is realized with a web-services approach. Each interface of our system is accessible by using the SOAP protocol (Eberhart & Fischer, 2002).

Related work

A kind of scheduling has been added lately in platforms like Blackboard or WebCT by the means of timed release of the resources. However, this cannot be compared to the use of pedagogical scenarios since the definition of the release time is done only for learning resources and does not take into account activities. Moreover, the time is defined beforehand and does not take into account the unfolding of the activities while COW provides activities and resources when they are needed according to the scenario and can manage time constraints relative to the start of the process or activities.

There are few platforms that provide the capability to enact pedagogical scenarios the Flexible e-learning system (Flex-eL) (Lin *et al.*, 2001) from Distributed Systems Technology Centre (DSTC) and the university of Queensland in Australia, the Virtual Campus Project developed at Politecnico di Milano (Cesarini *et al.*, 2004) and CopperCore from Open Universiteit Nederland (http://www.coppercore.org/).

Flex-eL consists of a distance education platform supported by a flexible workflow system. There is a learning process for each student. This allows an easy way to adapt the learning path to the real need. Groups of students are dynamically constructed. Managed Workflows are sequential and time constraints are not taken into account. The Virtual Campus project relies on the BizTalk business process engine from Microsoft. Hence scenario designers can take benefit from the capabilities of the XLANG language (Thatte, 2001) to express the learning activities organization.

Coppercore is an implementation of an IMS-LD interpretation engine. It is based on similar technological basis as COW. COW is halfway between CopperCore and the other projects since it is based on Workflow rather than a dedicated Educational Modelling Language but we are working on an import mechanism to support IMS-LD.

Interactive Study Guide

So as to illustrate the potential of the Workflow engine as a means to support pedagogical scenarios and show how we can provide indicators about the learners' progress within a course, we have built a user interface called the *Interactive Study Guide*. This learner oriented environment provides the list of the activities he has to perform according to the pedagogical scenario as well as indicators of is progression relative to the allocated

time and to the group progression. The next sections show how we can give indicators to help the learner plan his work.

Global view and indicators

Figure 10 shows the global view presented to the learner. It is composed of indicators and elements for easy navigation through the activities.

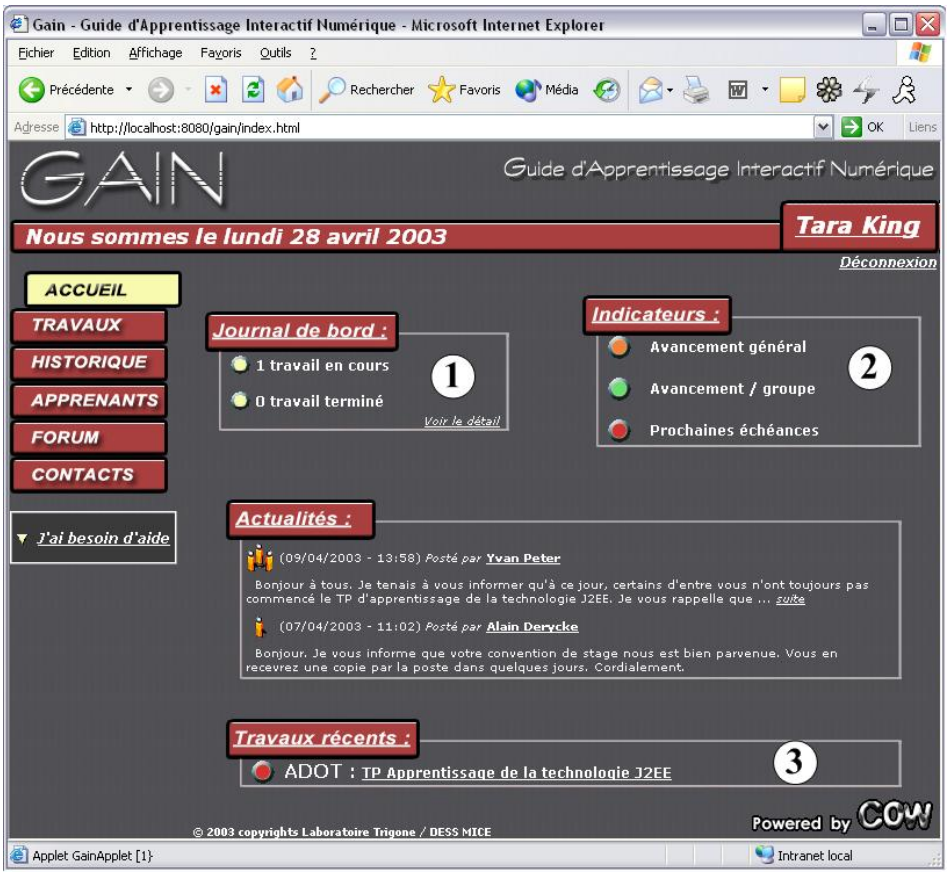


Figure 10. Global view and indicators

- **The log book** ① which shows the activities terminated and in progress so as to easily check if the work plan is respected.
- **Indicators** ② to know the progression within the course, relative to the group and relative to the next deadlines.
- **Recent works** ③ to access directly the last activities without having to navigate through the scenario.

Unit of study indicators and navigation

Figure 11 shows how the course can be presented to the learner. The interface is composed of the following elements.

- **Header** which gives the title of the course ①, a description of the pedagogical objectives ②, deadline for the termination of the course ③ as well as an indicator of the progression within the course ④.
- **Activity sections** which compose the course. Sections can be further broken up into arbitrary levels. For each activity there is a description, the estimated time and actual time spent on the activity ⑤ and the possibility to start/suspend/stop the activity ⑥.

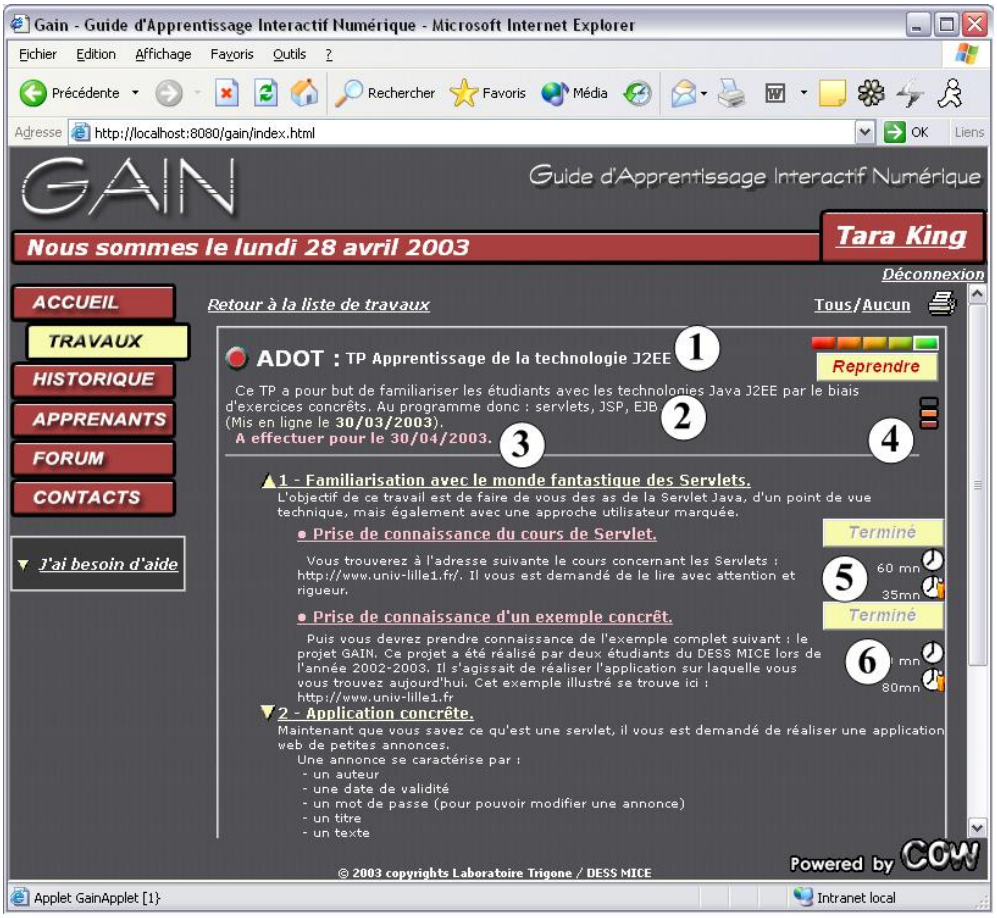


Figure 11. Unit of study indicators

Building indicators

To be able to construct the indicators, we have added a notion of estimated time for completion of the activity in the model. By this way, we are able to compare the actual time spent with the estimated time so as to tell the learner if he is late or in advance at the level of the course or the single activity. The group indicator is built by situating the learner relative to the average progression of the group. Finally for each activity for which there is a deadline, we calculate the remaining time so as to raise attention of the learner towards these activities.

Conclusion

Pedagogical scenarios raise great interest within the Technology Enhanced Learning field. They can be at the centre of the engineering of the courses and are a means to define the use of tools and learning objects during the course as well as the tasks in which people are involved. Introducing pedagogical scenarios and activities in Learning Management Systems must be handled at both organisational level and technical level. Multiple actors and phases are implied in the definition and operation of scenarios. The pedagogical engineer is a key actor since he has both pedagogical and technical competencies and can bridge the gap between the two worlds. Indeed, the production of pedagogical scenarios is not an easy task as no clear methodology exists at the moment. For this reason, initiatives such as the UNFOLD European project are created to support communities of interest and share knowledge (see http://www.unfold-project.net/UNFOLD). To lower the cost of this production, the scenarios must have a high reusability and be easily adapted. For this reason the scenario must not be too much tied to actual learning objects and tools which should be chosen at deployment and/or instantiation time according to the platform and learners. Continuous adaptation of the learning scenarios (i.e., during enactment of the scenario) is also important to provide the best result for a specific group of learners.

Starting from the idea that executing a pedagogical scenario is not so far from executing a process on a Workflow engine, we have developed a flexible Workflow engine suited to the flexible execution of pedagogical

scenarios. Based on a sample course in physics, we show how we can model a pedagogical scenario using a modified version of the standard XML Process Definition Language. This modified version enables the modelling of individual and group work within a process and the definition of time constraints. The engine allows the modification of the model at run-time for smooth adaptation and keeps track of the modified models for easy reuse of the enhanced models.

Having a definition of a pedagogical scenario with the different activities to perform can be used to provide support for the learners in their planning and to understand how they will reach a pedagogical objective of a unit of study. For this, we have developed an *Interactive Study Guide*, a user interface which helps tutors and learners plan their work with the help of indicators of the advancement within a unit of study and relative to the group.

Since the beginning of this work, IMS-LD has emerged as a standard language for describing pedagogical scenarios. Recently a first engine, *CopperCore* (CopperCore), has become available to interpret IMS-LD models and some platforms like *EduPlone* (http://eduplone.net) are starting to support it. This language seems more comprehensible by pedagogical engineer to formalize the scenarios. However, IMS-LD suffers some deficiencies like the scheduling model which is almost sequential (Caeiro *et al.*, 2003) or the missing management of the data-flow. We aim now to use the experience we have gained on Workflow systems to support in a flexible way an augmented version of IMS-LD.

References

Caeiro, M., Anido, L., & Llamas, M. (2003). A Critical Analysis of IMS Learning Design. In B. Wasson, S. Ludvigsen & U. Hoppe (Eds), *Proceedings of the International Conference on Computer Support for Collaborative Learning*, Dordrecht: Kluwer Academic Publishers, 363-367.

Cesarini, M., Monga, M., & Tedesco, R. (2004). Carrying on the elearning process with a workflow management engine. In H. Haddad, A. Omicini, R. L. Wainwright, & L. M. Liebrock (Eds.), *Proceedings of the 2004 ACM Symposium on Applied Computing (SAC)*, New York: ACM, 940-945.

Dillenbourg, P. (2002). Over-scripting CSCL: the risks of blending collaborative learning with instructional design. In P. A. Kirschner (Ed.). *Three worlds of CSCL. Can we support CSCL*, Heerlen: Open Universiteit Nederland, 61-91.

Eales, R. T. J., Hall, T., & Bannon, L. J. (2002). The Motivation is the Message: Comparing CSCL in different Settings. In: G. Stahl (Ed.), *Computer Supported Collaborative Learning: Foundations for a CSCL Community*, Hillsdale, NJ: Lawrence Erlbaum Associates, 310-317.

Eberhart, A., & Fischer, S. (2002). *Java Tools: Using XML, EJB, CORBA, Servlets and SOAP*, New York: Wiley.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns: Elements of reusable Object-Oriented Software*, Reading, MA: Addison-Wesley.

Grob, H. L., Bensberg, F., & Dewanto, B. L. (2004). Developing, Deploying, Using and Evaluating an Open Source Learning Management System. *Paper presented at the 26th International Conference on Information Technology Interfaces*, June 7-10, 2004, Cavtat, Croatia.

Hollingsworth, D. (1995). *The Workflow Reference Model - version 1.1*, retrieved July 25, 2005, from http://www.wfmc.org/standards/docs/tc003v11.pdf.

Hummel, H., , Manderveld, J., , Tattersall, C., & Koper, R. (2004). Educational modelling language and learning design: new opportunities for instructional reusability and personalised learning. *International Journal on Learning Technology, 1* (1), 111-126.

IMS (2003). *IMS Learning Design Information Model - version 1.0*, retrieved July 25, 2005, from http://www.imsglobal.org/learningdesign/index.html.

Jacobson, I., Booch, G., & Rumbaugh, J. (1999). *The Unified Software Development Process*, Reading, MA: Addison-Wesley.

Kiczales, G., des Rivières, G., J., & BoBrow, D. G. (1991), *The Art of the Metaobject Protocol*, Cambridge, MA: MIT Press.

Lin, J., Ho, C., Sadiq, W., & Orlowska, M. E. (2001). On Workflow Enabled e-Learning Services. In T. Okamoto, R. Hartley, Kinshuk, & J. P. Klus (Eds.), *Proceedings of the IEEE International Conference on Advanced Learning Technologies: Issues, Achievements and Challenges*, Los Alamitos, CA: IEEE Computer Society, 349-352.

Matena, V., & Stearns, B. (2001). *Applying Enterprise JavaBeans: Component-Based Development for the J2EE platform*, Reading, MA: Addison-Wesley.

OMG (2000). *Workflow Management Facility - Version 1.2*, retrieved July 25, 2005, from http://www.omg.org/docs/formal/00-05-02.pdf.

Schmidt-Wesche, B. (2003). *IBM WebSphere Platform User Roles*, retrieved July 25, 2005, from http://www-106.ibm.com/developerworks/websphere/library/techarticles/0303_schmidt/schmidt.html.

Thatte, S. (2001). *XLANG – Web Services for Business Process Design*, retrieved July 25, 2005, from http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm.

Vantroys, T., & Peter, Y. (2003). COW, a Flexible Platform for the Enactment of Learning Scenarios. *Lecture Notes in Computer Science, 2806*, 168-182.

WFMC (2002). *Workflow Process Definition Interface - XML Process Definition Language - version 1.0*, retrieved July 25, 2005, from http://www.wfmc.org/standards/docs/TC-1025_10_xpdl_102502.pdf.